

## PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-306038  
 (43)Date of publication of application : 05.11.1999

(51)Int.CI. G06F 9/46  
 G06F 15/16

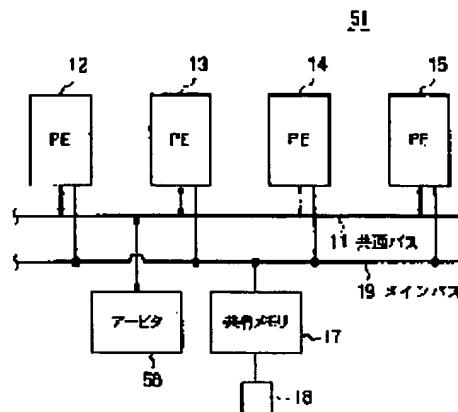
(21)Application number : 10-106756 (71)Applicant : SONY CORP  
 (22)Date of filing : 16.04.1998 (72)Inventor : IMAMURA YOSHIHIKO

## (54) UNITS AND METHOD FOR PARALLEL OPERATION PROCESSING

## (57)Abstract:

PROBLEM TO BE SOLVED: To provide a parallel operation units which can speedily and flexibly solve synchronization wait when plural tasks occur.

SOLUTION: A processor element PE 12 calls out plural tasks for PEs 13 to 15 and has a synchronization wait instruction performing synchronization wait as the necessity arises. An arbiter 56 increases a count value when task calling occurs and decreases the count value when the task is completed. The processor PE 12 compares the count value included in the synchronization wait instruction with the count value of the arbiter 56 when the synchronization waiting instruction is executed, releases the synchronization wait when the values coincide and performs the synchronization wait when they do not.



## LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平11-306038

(43)公開日 平成11年(1999)11月5日

(51) Int.Cl.<sup>6</sup>

G 06 F 9/46  
15/16

識別記号

3 6 0  
4 3 0

F I

G 06 F 9/46  
15/16

3 6 0 F  
4 3 0 B

審査請求 未請求 請求項の数11 O L (全 10 頁)

(21)出願番号 特願平10-106756

(22)出願日 平成10年(1998)4月16日

(71)出願人 000002185

ソニー株式会社

東京都品川区北品川6丁目7番35号

(72)発明者 今村 義彦

東京都品川区北品川6丁目7番35号 ソニー株式会社内

(74)代理人 弁理士 佐藤 隆久

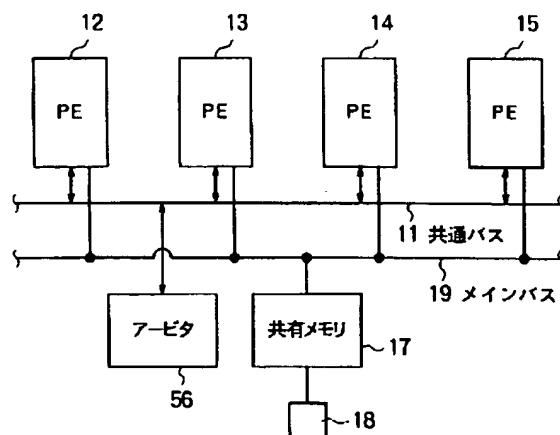
(54)【発明の名称】 並列演算処理装置およびその方法

(57)【要約】

【課題】 複数のタスクを発生した場合の同期待ちを高速かつ柔軟に解決できる並列演算処理装置を提供する。

【解決手段】 プロセッサエレメントPE12がPE13～15に対して複数のタスクを呼び出し、同期待ち命令で必要に応じて同期待ちを行う。アビタ56は、タスク呼び出しが発生するとカウント値を増加し、そのタスクが終了するとカウント値を減少させる。プロセッサエレメントPE12は、同期待ち命令を実行したときに、同期待ち命令に含まれるカウント値と、アビタ56のカウント値とを比較し、一致した場合に同期待ちを解除し、一致しない場合に同期待ちとなる。

51



## 【特許請求の範囲】

【請求項1】 単数または複数のタスク呼び出し命令を実行した後に、同期待ち命令で必要に応じて同期待ちを行う第1の演算処理手段と、

前記第1の演算処理手段から呼び出されたタスクを実行し、当該呼び出されたタスクが終了したときにタスク終了命令を実行する単数または複数の第2の演算処理手段と、

前記第1の演算処理手段による前記タスク呼び出し命令の実行に応じてカウント値を増加し、前記第2の演算処理手段の前記タスク終了命令の実行に応じてカウント値を減少するカウント手段とを有し、

前記第1の演算処理手段は、前記同期待ち命令に含まれるカウント値と、前記カウント手段のカウント値とを比較し、当該比較の結果に応じて同期待ちを解除するか否かを決定する並列演算処理装置。

【請求項2】 前記第1の演算処理手段は、前記同期待ち命令に含まれるカウント値と、前記カウント手段のカウント値とが一致したときに、同期待ちを解除する請求項1に記載の並列演算処理装置。

【請求項3】 前記同期待ち命令に含まれるカウント値は、前記第1の演算処理手段から呼び出されているタスクの数に比べて小さい請求項2に記載の並列演算処理装置。

【請求項4】 前記第1の演算処理手段における処理および前記単数または複数の第2の演算処理手段における処理とは相互に独立して行われる請求項1に記載の並列演算処理装置。

【請求項5】 前記同期待ち命令は、前記カウント値を引数として持つ請求項1に記載の並列演算処理装置。

【請求項6】 前記第1の演算処理手段および前記単数または複数の第2の演算処理手段は、共通のバスを介して接続されている請求項1に記載の並列演算処理装置。

【請求項7】 第1の演算処理において、単数または複数のタスク呼び出し命令を実行し、単数または複数の第2の演算処理において、前記第1の演算処理から呼び出されたタスクを実行し、当該呼び出されたタスクが終了したときにタスク終了命令を実行し、

前記第1の演算処理による前記タスク呼び出し命令の実行に応じて第1のカウント値を増加し、前記第2の演算処理の前記タスク終了命令の実行に応じて前記第1のカウント値を減少し、

前記第1の演算処理において、同期待ち命令が実行されたときに、前記同期待ち命令に含まれる第2のカウント値と、前記第1のカウント値とを比較し、当該比較の結果に応じて同期待ちを解除するか否かを決定する並列演算処理方法。

【請求項8】 前記第1の演算処理は、前記同期待ち命令に含まれる第2のカウント値と、前記第1のカウント値

とが一致したときに、同期待ちを解除する請求項7に記載の並列演算処理方法。

【請求項9】 前記同期待ち命令に含まれる第2のカウント値は、前記第1の演算処理から呼び出されているタスクの数に比べて小さい請求項8に記載の並列演算処理方法。

【請求項10】 前記第1の演算処理における処理および前記単数または複数の第2の演算処理における処理とは相互に独立して行われる請求項7に記載の並列演算処理方法。

【請求項11】 前記同期待ち命令は、前記カウント値を引数として持つ請求項7に記載の並列演算処理方法。

## 【発明の詳細な説明】

## 【0001】

【発明の属する技術分野】 本発明は、複数のタスクを発生した場合の同期待ちを高速かつ柔軟に解決できる並列演算処理装置およびその方法に関する。

## 【0002】

【従来の技術】 例えば、複数のプロセッサエレメント

20 (Processor Element: PE) が独立したプログラムカウンタを持ち、共通バスを介して相互に通信を行いながら処理を実行するMIMD (Multiple Instruction Multiple Datastream)型のマルチプロセッサシステムが知られている。このようなマルチプロセッサシステムは、コンカレント (並行) マルチタスクを行うことを前提としており、タスクを生成しようとしているメインプログラムを実行しているプロセッサエレメントと、新たなタスクが生成されるプロセッサエレメントとの間で通信を行う。このとき、タスクを呼んだ (生成した) プログラム

30 が、呼ばれた (生成された) タスクが終了するまで同期待ちを行う場合がある。

【0003】 図9は、一般的なマルチプロセッサシステム1の全体構成図である。図9に示すように、マルチプロセッサシステム1は、4つのプロセッサエレメントPE12, PE13, PE14, PE15と、タスクの同期を管理するアビタ16とが共通バス11を介して接続されている。共通バス11は、プロセッサエレメントPE12～15の相互間でコマンドなどの制御信号を送受信するための制御線として機能する。

40 【0004】 また、マルチプロセッサシステム1では、プロセッサエレメントPE12, PE13, PE14, PE15と共有メモリ17とがメインバス19を介して接続されている。共有メモリ17は、外部端子18を介して、外部メモリ (メインメモリ) に接続されている。なお、マルチタスクの同期を実現するマルチプロセッサシステムの構成としては、図9に示す構成以外にも種々のものがある。例えば、図9に示す例では、アビタ16でタスクの同期が集中管理される場合を示したが、例えば、アビタ16を設けずに、個々のプロセッサエレメントPE12～15にタスクの同期を管理する機能を

持たせてもよい。

【0005】図10は、タスクを生成するプログラム（メインプログラム25）が同期待ちを行う手法を説明するための図である。図10に示す例では、プロセッサエレメントPE12上で動作するメインプログラム25が、プロセッサエレメントPE13上にタスク26を生成している。プロセッサエレメントPE12および13は、個々のプロセッサ固有のマシン語（機械語）で記述された命令を実行して動作する。なお、タスクの生成ならびに同期の解決は、ハードウェアによる順序回路を用いても可能である。なお、本明細書では、命令によって同期機能の実現する場合を主に説明する。

#### 【0006】

【発明が解決しようとする課題】ところで、従来のマルチプロセッサシステムでは、以下に示す理由により、図10に示すメインプログラム25から複数のタスクを任意の数だけ生成することが困難であった。すなわち、マルチプロセッサシステムは、コンカレントマルチタスクを実行するが、このマルチタスク方式では、複数のプログラム（タスク）を複数のプロセッサエレメントPEに割り当てる必要がある。ここで、シングルプロセッサを前提としたマルチタスクでは、TSS（Time Sharing System）方式のように、一つのプロセッサエレメントPEに対して時分割で複数のタスクを割り当てることが最も一般的である。従って、一つのプロセッサエレメントPEを中心としてタスク管理テーブルを一組だけ用意すればよい。このTSS方式を採用する多くの場合、それらタスクの切り替え機構を持つUnix（商標名）などのOS（Operating System）を用いることになる。

【0007】通常、プロセッサエレメントPEは、マルチタスクを特に意識した同期命令を用意していないことが多い。同期命令を用いるよりも、タイマなどの外部割り込みイベントを通じて例外処理を発生させ、その結果、タスクの切り替えを行う方式を採用することが多い。また、タスクの切り替えをより高速に実行するために、プロセッサエレメントPE内でハードウェアによるサポートを行っていることが多いが、基本的にはソフトウェアによってタスク切り替え機能が実現される。

【0008】これに対して、マルチプロセッサシステムでは、前述したTSS方式を採用する場合、複数のタスク管理テーブルを用意することが必要になる。また、それら複数のタスク管理テーブルを総合的に管理するプログラムを、個々のプロセッサエレメントPEを管理するプログラムの一階層上に用意する必要があり、かなり複雑なOSになる。そのため、従来のマルチプロセッサシステムでは、図10に示すメインプログラム25から複数のタスクを任意の数だけ生成することが困難であった。なお、マルチプロセッサシステムに実装するOSは、通常、そのマルチプロセッサシステムを使うサイド（ユーザ）が決定する。

【0009】ところで、マルチタスクの実現方法は、TSS方式の他にも存在する。例えば、特定のプロセッサエレメントPEをコプロセッサとして用いるなどのやや特殊用途への応用が考えられる。その他にも、特定のプロセッサエレメントPEをコプロセッサとして固定しないまでも、コプロセッサで実行するプログラムを常駐させることも、ある分野では非常に有効である。いずれにしても、マルチプロセッサシステムには、タスクの同期機構が必要である。

10 【0010】研究試作段階のマルチプロセッサシステムでは、一般的に、個々のプロセッサエレメントPE毎にシングルプロセッサと同じOSを搭載している。そして、これらプロセッサエレメントPE間の通信を行うことにより、全体としてマルチタスクを達成していることが多い。この場合、プロセッサエレメントPE相互の通信の機能の一部に同期機構が用いられる。あるいは、セマフォアなどのメモリを介した同期機構も採用可能である。

【0011】しかしながら、タスクの生成とそのタスクの同期待ちに関しては、いずれの場合にも最終的にはソフトウェアによる処理を行っているのでレスポンスが悪く、せいぜい粗粒度並列プログラムを実行する場合に応用されるのが現状である。また、仮に、複数のタスクを生成することが十分可能なシステムであっても、それらのタスクの終了を待つ（同期待ち）方式に決定的な解決手段がない。

20 【0012】メインプログラムから生成された複数のタスクのうちどのタスクを待つかなどの条件設定は、ソフトウェアによるプログラム記述によれば全ての組み合わせが可能であるが、それらの条件判定に費やす時間のオーバーヘッドもかなり大きくなり、高速に同期をとることができない。一方、このような設定条件を、ハードウェアによって決定するものがある。例えば、米国のインテル社が開発したマイクロプロセッサ8086とそのプロセッサ専用に設計されたコプロセッサ8087との間では、ハンドシェークによる同期待ちシステムが確立されている。プロセッサ8086上のメインプログラムの数値演算用の命令を実行すると、自動的にコプロセッサ8087はその命令を解釈して演算を始める。通常、数値演算の実行には複数のクロックサイクルが必要とされていた。従って、その間、プロセッサ8086では当該命令の次の命令を順次実行する。

30 【0013】メインプログラムには、タスクの発生命令から適当な命令数を経た後に同期命令が記述される。そして、その同期命令が実行される前に当該数値演算が終了していれば、プロセッサ8086は数値演算が同期されているものとみなしそのまま命令実行を進める。また、その同期命令が実行される前に当該数値演算が終了していなければ、コプロセッサ8087の演算が終了するまで同期待ちをする。この同期待ちシステムは、簡単

なプロトコルによるハンドシェーク信号を用いており、きわめて簡単な構成で高速に同期をとることができる。しかしながら、複数のコプロセッサ8087をひとつのプロセッサ8086に接続することができず、複数のタスクを生成した場合の同期待ちを解決できないという問題がある。

【0014】本発明は上述した従来技術に鑑みてなされ、複数のタスクを発生した場合の同期待ちを高速かつ柔軟に解決できる並列演算処理装置およびその方法を提供することを目的とする。

【0015】

【課題を解決するための手段】本発明は、前述した従来の同期待ち機構に改良を加えて、マルチタスクシステムの全体的な性能を向上させるものである。先に記述したマルチプロセッサシステムにおける同期機構において、メインプログラムから生成されたタスク（以下、子タスクとも記す）の終了を認識する際に要する時間が長いという問題があったが、これは、従来のマルチプロセッサシステムが汎用性を重視するあまり、ソフトウェアによる解決方法を選択したことに原因がある。

【0016】本発明は、タスクの生成とその同期に伴う汎用性をある程度制限している。タスクの生成はほぼ自由に任意の数分を許すものとし、それら子タスクの稼動状況／終了を自動的に数値（カウント値）に換算する。メインプログラムでは、同期命令の実行条件（成立条件）にその数値を含める。そして、子タスクの数値化された稼動状況と同期命令を組み合わせて用いる。これは、複数のタスクの生成を可能とする。また、同期命令は、子タスクの終了をハードウェアによって認識させる方法を採用し、レスポンスの高速化を図る。

【0017】ここで、本発明を実現する手段として、子タスクを生成する毎にメインプログラムを実行しているプロセッサエレメントは、生成した子タスクの数を記憶しておくことを提案する。これは、単にレジスタと加減算器とを組み合わせてもよいし、カウンタを用意するだけでもよい。カウンタを例にとると、初期値として0をセットしておき、子タスクを生成する毎に1だけインクリメント（増加）させていく。そして、子タスクが終了すると、しかるべき手続きを経てメインプログラムを実行しているプロセッサエレメントPEにその旨を通知して、先のカウント値を1だけ減ずる。

【0018】メインプログラムのなかの同期命令を実行するプロセッサエレメントは、同期命令の実行時に前述したカウント値とその同期命令に付加された引数の値とを比較する。比較の結果、カウント値のほうが小さいかあるいは同じ値であれば同期条件が成立したものとみなし、メインプログラムはその同期命令以降の命令実行に進む。もし、そうでなければ（カウント値のほうが大きければ）、同期条件を満たすまで同期待ちする。

【0019】同期命令実行後は、次に子タスクの生成に

備えて0に初期化をすることにしてもよい。また、そうしなくてもよい。これはそのシステムを使用する利用者の課題である。また、カウンタを設ける位置は特に限定されない。例えば、メインプログラムを実行するプロセッサエレメント内に存在させるか、あるいは、アビタモジュール内に存在させてもよい。

【0020】従来の同期機構では、仮に複数の子タスクを生成したときには、ソフトウェアによりすべての詳細な同期条件を記述するか、あるいは、ハードウェアによりすべての子タスクの終了を待つかの方法を採用していた。これに対して、本発明は、子タスクの終了をハードウェアによって認識させるわけであるが、同期待ち命令に伴う同期条件に制限を付加して、単に同期させるべき子タスクの数のみを設定する。これにより、同期機構にある程度の柔軟性を持たせながら論理回路の規模削減を図ることができる。

【0021】すなわち、本発明の並列演算処理装置は、単数または複数のタスク呼び出し命令を実行した後に、同期待ち命令で必要に応じて同期待ちを行う第1の演算処理手段と、前記第1の演算処理手段から呼び出されたタスクを実行し、当該呼び出されたタスクが終了したときにタスク終了命令を実行する単数または複数の第2の演算処理手段と、前記第1の演算処理手段による前記タスク呼び出し命令の実行に応じてカウント値を増加し、前記第2の演算処理手段の前記タスク終了命令の実行に応じてカウント値を減少するカウント手段とを有し、前記第1の演算処理手段は、前記同期待ち命令に含まれるカウント値と、前記カウント手段のカウント値とを比較し、当該比較の結果に応じて同期待ちを解除するか否かを決定する。

【0022】また、本発明の並列演算処理装置は、好ましくは、前記第1の演算処理手段は、前記同期待ち命令に含まれるカウント値と、前記カウント手段のカウント値とが一致したときに、同期待ちを解除する。

【0023】また、本発明の並列演算処理装置は、好ましくは、前記同期待ち命令に含まれるカウント値は、前記第1の演算処理手段から呼び出されているタスクの数に比べて小さい。

【0024】さらに、本発明の並列演算処理方法は、第40 1の演算処理において、単数または複数のタスク呼び出し命令を実行し、単数または複数の第2の演算処理において、前記第1の演算処理から呼び出されたタスクを実行し、当該呼び出されたタスクが終了したときにタスク終了命令を実行し、前記第1の演算処理による前記タスク呼び出し命令の実行に応じて第1のカウント値を増加し、前記第2の演算処理の前記タスク終了命令の実行に応じて前記第1のカウント値を減少し、前記第1の演算処理において、同期待ち命令が実行されたときに、前記同期待ち命令に含まれる第2のカウント値と、前記第1のカウント値とを比較し、当該比較の結果に応じて同期

待ちを解除するか否かを決定する。

【0025】

【発明の実施の形態】以下、本発明の実施形態に係わるマルチプロセッサシステムを説明する。図1は、本実施形態のマルチプロセッサシステム51の構成図である。図1に示すように、マルチプロセッサシステム51は、4つのプロセッサエレメントPE12, PE13, PE14, PE15と、タスクの同期を管理するアービタ56とが共通バス11を介して接続されている。また、マルチプロセッサシステム51では、プロセッサエレメントPE12, PE13, PE14, PE15と共有メモリ17とがメインバス19を介して接続されている。共有メモリ17は、外部端子18を介して、外部メモリ(メインメモリ)に接続されている。

【0026】図1において、図9と同じ符号を付したプロセッサエレメントPE12, PE13, PE14, PE15、共通バス11、メインバス19、共有メモリ17および外部端子18は、前述した図9に示す構成要素と同じである。すなわち、マルチプロセッサシステム51は、アービタ56におけるタスク同期の管理方法に特徴を有する。アービタ56は、カウンタを備え、例えば、プロセッサエレメントPE12が子タスクの生成を伴う命令を実行したときに、カウンタのカウント値を1だけ増加し、子タスクが終了したときにカウントのカウント値を1だけ減少させる。

【0027】マルチプロセッサシステム51は、プロセッサエレメントPE12～15が、他のプロセッサエレメントPEに対して任意に子タスクを生成することができる。子タスクはそのプログラムの終了時に、当該子タスクを呼び出した親のプログラムに対して、子タスクの終了を示すメッセージを送信する。このメッセージは、図1の共通バス11を用いて行われるものとするが、その形態はいかなるものであってもよい。プロセッサ間通信を厳密に規定してもよいし、単に信号線を通じて知らせてよい。

【0028】なお、説明の都合上、子タスクとして生成されるプログラムのアドレス(メインメモリ上の番地)は予めメインプログラムの記述のなかで設定されているものとする。また、子タスクは、プロセッサエレメントPE12以外のプロセッサエレメントPE13～15上に生成されるものとする。具体的にどのプロセッサエレメントPEに生成するかは図1に示すアービタ56が自動的に決定する。本実施形態では、子タスクの割り当て方法について触れない。

【0029】子タスクを生成する命令としては、例えば、「gen」命令を用いるが、その名前は任意である。従って、「gen」命令がブリミティブなマシン語ではなく、アセンブラー言語のマクロ命令であっても良い。子タスクのプログラムの中で、タスク生成やその同期、終了に関するもの以外の通常の命令を「inst

」、「inst2」と記述する。これも前記と同様に、それら命令の名前や構成方法は任意である。また、子タスクの終了命令を「end」とする。この命令によって、子タスクを呼んだプロセッサエレメントPEに対して、タスクの終了を示すメッセージを自動的に送る。子タスクで作成したデータその他の情報は、適当に処理するものとする。また、メインプログラムにおいて、同期待ちのための命令を「wait」とする。後述するが、この「wait」命令には引数を設定する事ができる。

【0030】以下、マルチプロセッサシステム51の動作を説明する。単数の子タスクのみを生成する場合図2はマルチプロセッサシステム51において一つの子タスクを生成する場合のプロセッサエレメントPE上で動作するプログラムを説明するための図、図3は図2に示す場合におけるタスク発生および同期待ち解除のタイミングを説明するための図である。ここで、メインプログラム25の「wait」命令は、アービタ56のカウンタのカウント値が「0」になることを同期解除条件としている。例えば、図3に示すタイミング「r1」で、プロセッサエレメントPE12において、図2に示すメインプログラム25に含まれる「gen」命令が実行され、タイミング「n1」でプロセッサエレメントPE13上にタスク26が生成される。このとき、アービタ56のカウント値が「1」に設定される。そして、プロセッサエレメントPE13において、タスク26の「inst1」および「inst2」などの命令が実行される。また、プロセッサエレメントPE12において、メインプログラム25の「gen」命令以降の命令が実行される。そして、図3に示すタイミング「s1」で、プロセッサエレメントPE12において、図2に示す「wait」命令が実行される。このとき、アービタ56のカウント値が「1」であるため、プロセッサエレメントPE12は、同期条件が満たされていないと判断し、同期待ち状態になる。

【0031】そして、図3に示すタイミング「s2」で、プロセッサエレメントPE13においてタスク26の「end」命令が実行されると、タスク26が終了すると共に、アービタ56がカウント値が1だけ減算されて「0」になる。これにより、同期条件が成立し、プロセッサエレメントPE12は、メインプログラム25の「wait」命令以降の命令を実行する。なお、図2および図3に示す場合において、プロセッサエレメントPE12が、メインプログラム25に含まれる「wait」命令が実行される前に、プロセッサエレメントPEがタスク26の「end」命令を実行した場合には、プロセッサエレメントPE12は同期待ち状態にならない。

【0032】複数の子タスクを生成する場合(その1)  
図4はマルチプロセッサシステム51において複数のタ

スクを生成し、全てのタスクの終了を同期待ちの条件とした場合の各プロセッサエレメントPE上で動作するプログラムを説明するための図、図5は図4に示す場合におけるタスク発生および同期待ち解除のタイミングを説明するための図である。図4および図5に示す例では、図4に示すメインプログラム75の「wait」命令が示す同期待ち解除条件がカウント値「0」となってい。この場合には、図5に示すタイミング「r1」で、プロセッサエレメントPE12において、図4に示すメインプログラム75に含まれる「gen1」命令が実行され、タイミング「r2」でプロセッサエレメントPE13上にタスク76が生成される。このとき、アービタ56のカウント値が「1」に設定される。また、図5に示すタイミング「r2」で、プロセッサエレメントPE12において、図4に示すメインプログラム75に含まれる「gen2」命令が実行され、タイミング「r3」でプロセッサエレメントPE14上にタスク77が生成される。このとき、アービタ56のカウント値が「2」に設定される。また、図5に示すタイミング「r3」で、プロセッサエレメントPE12において、図4に示すメインプログラム75に含まれる「gen3」命令が実行され、タイミング「n1」でプロセッサエレメントPE15上にタスク78が生成される。このとき、アービタ56のカウント値が「3」に設定される。

【0033】そして、プロセッサエレメントPE13において、タイミング「r2」から、タスク76に含まれる「inst1」および「inst2」などの命令が実行される。また、プロセッサエレメントPE14において、タイミング「r3」から、タスク77に含まれる「inst1」および「inst2」などの命令が実行される。また、プロセッサエレメントPE15において、タイミング「n1」から、タスク78に含まれる「inst1」および「inst2」などの命令が実行される。

【0034】そして、図5に示すタイミング「e1」で、プロセッサエレメントPE13においてタスク76の「end」命令が実行されると、タスク76が終了すると共に、アービタ56のカウント値が1だけ減算されて「2」になる次に、図5に示すタイミング「s1」で、プロセッサエレメントPE12において、図4に示す「wait」命令が実行される。このとき、アービタ56のカウント値が「2」であることから、同期待ち解除条件を満たさず、プロセッサエレメントPE12は同期待ち状態になる。

【0035】次に、図5に示すタイミング「e2」で、プロセッサエレメントPE15においてタスク78の「end」命令が実行されると、タスク78が終了すると共に、アービタ56がカウント値が1だけ減算されて「1」になる次に、図5に示すタイミング「s2」で、プロセッサエレメントPE14においてタスク77の

「end」命令が実行されると、タスク77が終了すると共に、アービタ56がカウント値が1だけ減算されて「0」になるこれにより、メインプログラム75の「wait」命令が示す同期待ち解除条件が満たされ、プロセッサエレメントPE12の同期待ちが解除される。

【0036】複数の子タスクを生成する場合（その2）

図6はマルチプロセッサシステム51において複数のタスクを生成し、2つのタスクの終了を同期待ちの条件とした場合の各プロセッサエレメントPE上で動作するプログラムを説明するための図、図7は図6に示す場合におけるタスク発生および同期待ち解除のタイミングを説明するための図である。図6および図7に示す例では、図6に示すメインプログラム85の「wait」命令が示す同期待ち解除条件がカウント値「1」となってい。この場合には、図7に示すタイミング「r1」で、プロセッサエレメントPE12において、図6に示すメインプログラム85に含まれる「gen1」命令が実行され、タイミング「r2」でプロセッサエレメントPE13上にタスク86が生成される。このとき、アービタ56のカウント値が「1」に設定される。また、図7に示すタイミング「r2」で、プロセッサエレメントPE12において、図6に示すメインプログラム85に含まれる「gen2」命令が実行され、タイミング「r3」でプロセッサエレメントPE14上にタスク87が生成される。このとき、アービタ56のカウント値が「2」に設定される。また、図7に示すタイミング「r3」で、プロセッサエレメントPE12において、図6に示すメインプログラム85に含まれる「gen3」命令が実行され、タイミング「n1」でプロセッサエレメントPE15上にタスク88が生成される。このとき、アービタ56のカウント値が「3」に設定される。

【0037】そして、プロセッサエレメントPE13において、タイミング「r2」から、タスク86に含まれる「inst1」および「inst2」などの命令が実行される。また、プロセッサエレメントPE14において、タイミング「r3」から、タスク87に含まれる「inst1」および「inst2」などの命令が実行される。また、プロセッサエレメントPE14において、タイミング「n1」から、タスク88に含まれる「inst1」および「inst2」などの命令が実行される。

【0038】そして、図7に示すタイミング「e1」で、プロセッサエレメントPE13においてタスク86の「end」命令が実行されると、タスク86が終了すると共に、アービタ56のカウント値が1だけ減算されて「2」になる次に、図7に示すタイミング「s1」で、プロセッサエレメントPE12において、図6に示す「wait」命令が実行される。このとき、アービタ56のカウント値が「2」であることから、同期待ち解除条件を満たさず、プロセッサエレメントPE12は同期待ち状態になる。

期待状態になる。

【0039】次に、図5に示すタイミング「e2」で、プロセッサエレメントPE15においてタスク88の「end」命令が実行されると、タスク88が終了すると共に、アビタ56がカウント値が1だけ減算されて「1」になる。これにより、メインプログラム75の「wait」命令が示す同期待ち解除条件が満たされ、プロセッサエレメントPE12の同期待ちが解除される。

【0040】このように、図6および図7に示す例では、図6に示すメインプログラム85が生成した3つの子タスク86、87、88のうち2つの子タスクの終了を「wait」命令の引数に記述する。その結果、タスク86および88が終了すると、タスク87の終了を待たずに同期待ち状態が解除できる。ここで、タスク87としては、例えば、仮想記憶をサポートするタスクなどの長期的に存在するタスクが適用される。

【0041】以上説明したように、マルチプロセッサシステム51によれば、マルチタスクを実現する場合でも、OSなどによる複雑な管理を行うことなく、複数のタスクを発生したプログラムの同期待ちを解決することができる。マルチプロセッサシステム51によるタスク管理は、ソフトウェアによるタスクの管理と比べてやや汎用性に制限が付くものの、同期待ち機構をハードウェアで実現することができるため容易になる。これは、タスク終了の事実を高速に知るための方法として有効であり、また、回路規模が小さく実現できるなどの効果がある。

【0042】また、同期待ち命令である「wait」命令に引数を記述することにより、メインプログラムと、当該メインプログラムが生成した子タスクとの間の同期条件を柔軟に設定できる。例えば、マルチプロセッサシステム51は、図6において、プロセッサエレメントPE14上に生成される子タスク87が他の子タスク86、88に比べてプログラムが終了するまでの時間が極めて大きくなることが事前に知ることができていて、なおかつ、プロセッサエレメントPE12上の同じメインプログラム85で子タスクを生成する必要性にせまられたときなどは特に有効である。

【0043】本実施形態では、意図的に極端な場合を想定している。しかし、タスクの処理時間に大小の差が存在することは一般的にいえることである。マルチプロセッサシステム上でマルチタスクを実現するためには、何らかの方法でプロセッサ資源と各々のスレッド（OS上の概念でタスクの断片のこと）とを対応づけることが求められる。プロセッサ資源を有効に使用するためには、システム上のプロセッサ稼動状況を的確に把握する必要があり、また、ある特定のプロセッサの動作状況によってシステム全体が影響を受けることはあまり望ましくはない。従来の技術を用いてマルチプロセッサシステム

ム上にマルチタスクを実現する場合には、前述したように、プロセッサ資源を有効に使用することが困難になる。つまり、ソフトウェアによる制御では稼動していないプロセッサに対して新たなタスクを割り当てることが可能であるが、その判断を下すまでに多くの時間を費やすことになる。結果的に、システム全体が遅く動作することになる。

【0044】これに対して、マルチプロセッサシステム51によれば、同期待ちの解決手段として、カウンタを用いるのみで、複雑なソフトウェアを用いたないため、同期待ちを高速に解決でき、リアルタイム性を向上できる。また、ハードウェアによる従来の技術では、子タスクの生成とその同期機構を高速にかつ簡単に構成することができる一方で、複数のタスクを生成することが困難になっている。仮に、複数のタスクが生成できたとしても、それまでに生成したタスクをすべて待つことになる。これに対して、マルチプロセッサシステム51によれば、「wait」命令に同期待ちを行うタスクの数を引数として設定することで、多様かつ柔軟な同期待ちを実現できる。

【0045】OSを実装する上で、新たなタスクを生成する際にはそれまで生成されているタスクのうちいくつまでが終了しているかなどを把握し、かつ、それらのタスクがどのプロセッサエレメントPEに割り当てられているかを知る必要がある。裏を返せば、すべてのタスクの終了を待っていては、新たなタスクのプロセッサエレメントPEへの割り当てができないことになるか、あるいは、相当の困難を伴うことになるだろう。従って、上述したマルチプロセッサシステム51のように、タスクを生成するべきメインプログラム内でタスクの終了状態を監視しておきその値を同期機構に反映させることができれば、マルチプロセッサのシステムを向上させることができる。

【0046】本発明は上述した実施形態には限定されない。上述した実施形態では、図1に示すプロセッサエレメントPE12がタスク生成命令である「gen」命令を実行する場合を例示したが、その他のプロセッサエレメントPE13～15が「gen」命令を実行してもよい。また、図1に示す例では、4個のプロセッサエレメントPE12～15を設けた場合を例示したが、2以上であればプロセッサエレメントPEの数は任意である。

【0047】上述した実施形態では、アビタ56にタスク同期機能を持たせた場合を例示したが、タスク同期機能をその他の構成要素に持たせてもよい。

【0048】また、本発明は、例えば、図8に示すように、コンピュータ102、103、104、105をネットワーク101を介して接続して分散処理を行う並列分散処理システムにも適用できる。この場合に、例えば、コンピュータ102がメインプログラムを実行する場合には、コンピュータ102内にアビタ106が設

けられる。アービタ106の機能は、前述した図1に示すアービタ56と同じである。

【0049】

【発明の効果】以上説明したように、本発明の並列演算処理装置によれば、第1の演算処理手段が複数のタスクを呼び出した場合の同期待ちを高速かつ柔軟に解決できる。また、本発明の並列演算処理方法によれば、第1の演算処理が複数のタスクを呼び出した場合の同期待ちを高速かつ柔軟に解決できる。

【図面の簡単な説明】

【図1】図1は、本発明のマルチプロセッサシステムの構成図である。

【図2】図2は、図1に示すマルチプロセッサシステムにおいて一つの子タスクを生成する場合のプロセッサエレメントPE上で動作するプログラムを説明するための図である。

【図3】図3は、図2に示す場合におけるタスク発生および同期待ち解除のタイミングを説明するための図である。

【図4】図4は、図1に示すマルチプロセッサシステムにおいて複数のタスクを生成し、全てのタスクの終了を同期待ちの条件とした場合の各プロセッサエレメントPE上で動作するプログラムを説明するための図である。

【図5】図5は、図4に示す場合におけるタスク発生および同期待ち解除のタイミングを説明するための図である。

【図6】図6は、図1に示すマルチプロセッサシステムにおいて複数のタスクを生成し、2つのタスクの終了を同期待ちの条件とした場合の各プロセッサエレメントPE上で動作するプログラムを説明するための図である。

【図7】図7は、図6に示す場合におけるタスク発生および同期待ち解除のタイミングを説明するための図である。

【図8】図8は、本発明を適用した並列分散処理システムの構成図である。

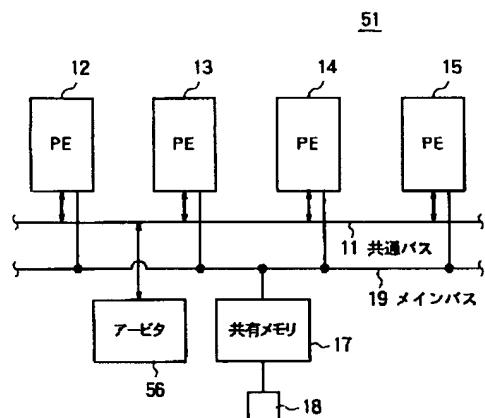
【図9】図9は、一般的なマルチプロセッサシステムの全体構成図である。

【図10】図10は、タスクを生成するプログラムが同期待ちを行う手法を説明するための図である。

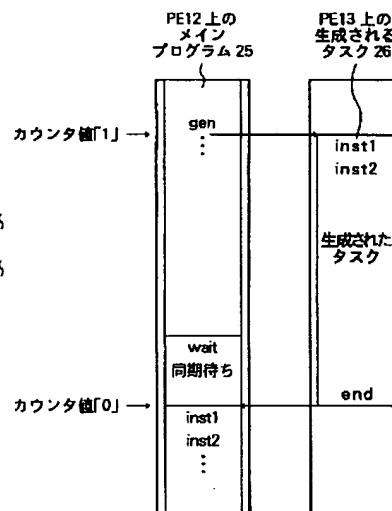
【符号の説明】

1, 51…マルチプロセッサシステム、11…共通バス、16, 56, 156…アービタ、17…共有メモリ、19…メインバス、12～15…プロセッサエレメントPE、18…外部端子、100…並列分散処理システム、102～105…コンピュータ

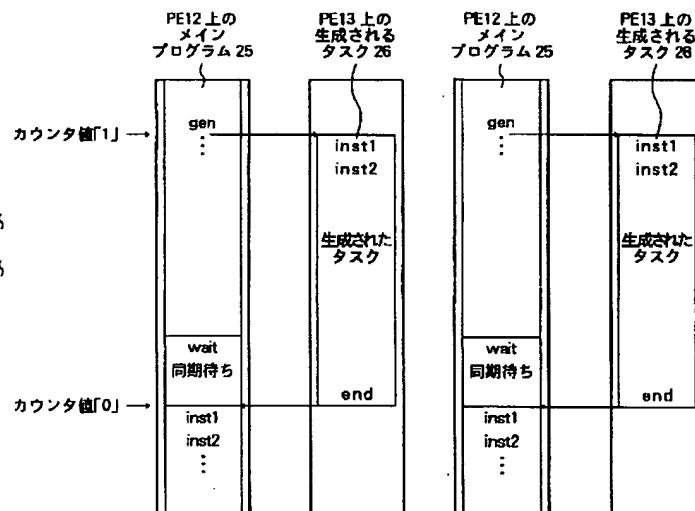
【図1】



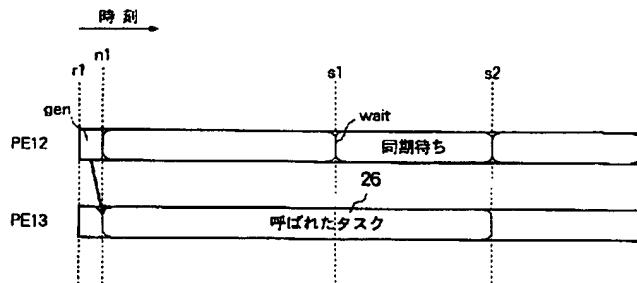
【図2】



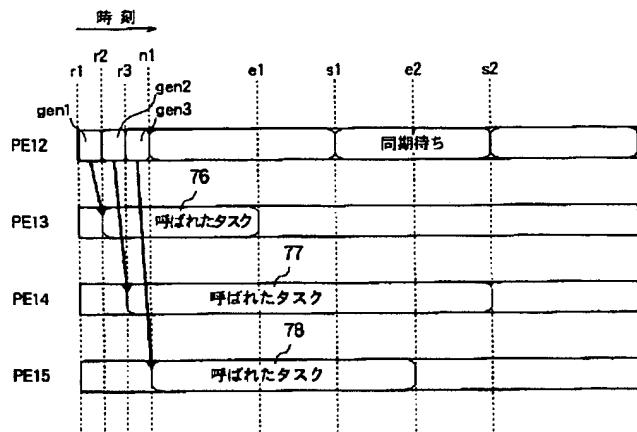
【図10】



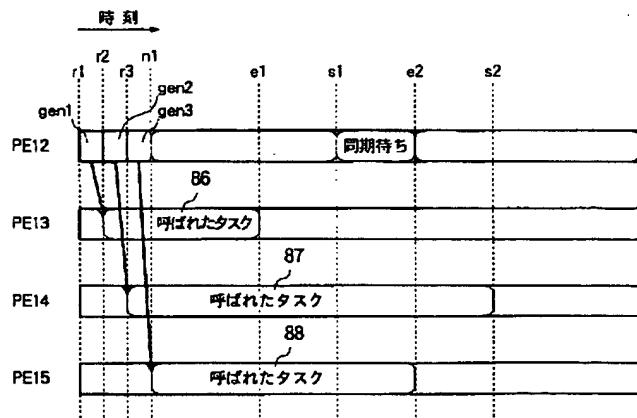
【図3】



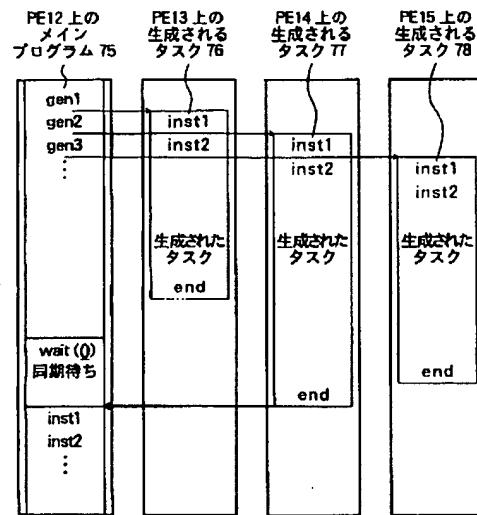
【図5】



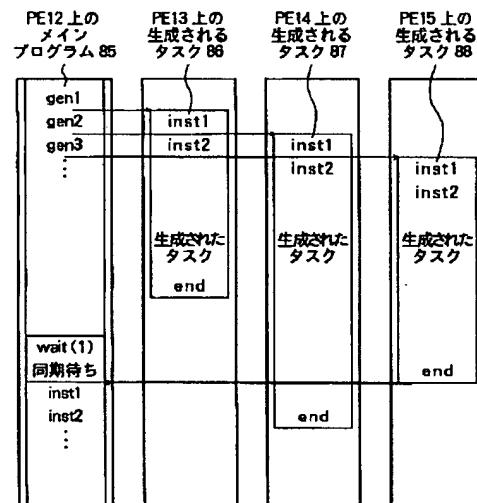
【図7】



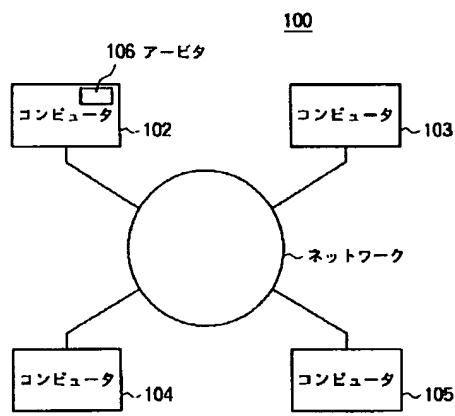
【図4】



【図6】



【図8】



【図9】

